How to Steer Your Adversary: Targeted and Efficient Model Stealing Defenses with Gradient Redirection

Mantas Mazeika [1] **Bo Li** [1] **David Forsyth** [1]

Abstract

Model stealing attacks present a dilemma for public machine learning APIs. To protect financial investments, companies may be forced to withhold important information about their models that could facilitate theft, including uncertainty estimates and prediction explanations. This compromise is harmful not only to users but also to external transparency. Model stealing defenses seek to resolve this dilemma by making models harder to steal while preserving utility for benign users. However, existing defenses have poor performance in practice, either requiring enormous computational overheads or severe utility trade-offs. To meet these challenges, we present a new approach to model stealing defenses called gradient redirection. At the core of our approach is a provably optimal, efficient algorithm for steering an adversary's training updates in a targeted manner. Combined with improvements to surrogate networks and a novel coordinated defense strategy, our gradient redirection defense, called GRAD^2, achieves small utility trade-offs and low computational overhead, outperforming the best prior defenses. Moreover, we demonstrate how gradient redirection enables reprogramming the adversary with arbitrary behavior, which we hope will foster work on new avenues of defense.

1. Introduction

As deep neural networks become more capable and economically valuable, responsibly democratizing access to the best available models could lead to widespread social good. Owners may not wish to just publish parameters, as this opens the door to malicious use and provides no financial return on investment (Radford et al., 2019; Buchanan et al., 2021). Prediction APIs have emerged as a solution to these problems, as they enable filtering out harmful use cases and support a software-as-a-service business model. Moreover, they allow users with minimal computational resources to access the largest available models at reasonable costs.

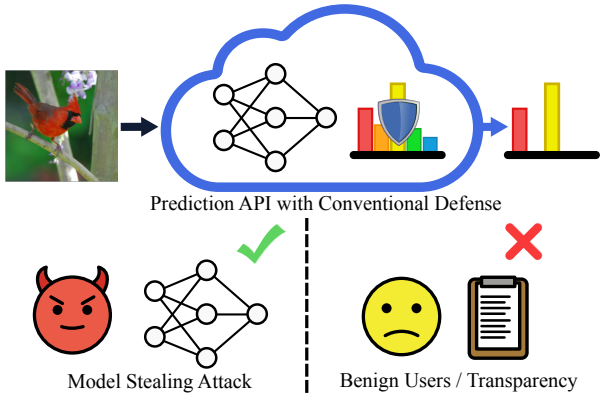

Figure 1. High-profile prediction APIs such as the OpenAI API and AI21 Studio truncate posteriors. This serves as a rudimentary defense against model stealing attacks, protecting financial investments. However, it also has significant trade-offs. Namely, it harms benign users who might otherwise benefit from the withheld predictions, and it reduces transparency.

The promise of prediction APIs is hampered by the fact that they are vulnerable to model stealing attacks. Malicious users can clone the functionality of an API by gathering a dataset of queries and responses and training their own model. This allows them to circumvent the expensive process of manual data curation, which can be a multi-million dollar investment for API providers (Hendrycks et al., 2021a; Tramèr et al., 2016). Additionally, model theft can be used as a stepping stone for mounting evasion attacks, which could result in critical failures in downstream applications utilizing the API (Papernot et al., 2017). In this paper, we investigate methods for defending against model stealing attacks.

In practice, prediction APIs often employ rudimentary obfuscation measures, such as truncating predicted posteriors to a minuscule fraction of their original size. For instance,

[1]UIUC. Correspondence to: Mantas Mazeika <mantas3@illinois.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

the OpenAI API and AI21 Studio truncate to 2% and 0.03% of the available information, respectively. While this can protect against model stealing, it has major side effects. As illustrated in Figure 1, truncating posteriors reduces external transparency and harms benign users who might otherwise benefit from the withheld predictions.

Recent developments in model stealing defenses provide an avenue towards enabling API providers to share more information about their models without fear of extraction attacks. The objective of model stealing defenses is to make models harder to steal without substantially altering posterior predictions. In recent years, several works have investigated perturbation-based defenses that seek to maximize adversary error while minimally altering posteriors. However, these defenses have poor performance in practice, as they either incur infeasible computational overheads or require large utility trade-offs to be effective.

To overcome these challenges, we propose a new approach to perturbation-based model stealing defenses, which we call gradient redirection. At the core of our approach is an efficient, provably optimal algorithm for steering an adversary's training updates. Unlike prior methods, gradient redirection enables altering the trajectory of an extraction attack in a targeted manner, enabling a wide range of possible defenses. Using gradient redirection, we develop an efficient defense called GRAD^2 that incorporates improved surrogate networks and a novel coordinated defense strategy. In extensive experiments, we find that GRAD^2 outperforms prior defenses across multiple threat models. Moreover, we show how gradient redirection enables reprogramming the adversary with arbitrary behavior, including hidden watermarks. Experiment code is available at `[Anonymized]`.

2. Related Work

Model Stealing Attacks. Numerous works have explored the vulnerability of prediction APIs to model extraction attacks, where the adversary's goal is to obtain a copycat network with similar functionality to the prediction API. In early work, Tramèr et al. (2016) identify a number of threat models and show that extraction of simple model classes is possible. For deep neural networks, Papernot et al. (2017) show that model extraction is possible and can facilitate subsequent evasion attacks. Both these works assume the ability to adaptively probe the API with optimized queries, which can be accurately detected in some cases by monitoring query patterns (Juuti et al., 2019; Pal et al., 2020). Thus, we focus our investigation on the case where adversary queries have no carefully crafted inputs or sequential structure. By leveraging knowledge distillation (Hinton et al., 2015), several works have shown that it is possible to steal the functionality of deep neural networks using weakly related queries (Orekondy et al., 2019) or even random queries (Krishna et al., 2019), although attacks with queries more related to the target task typically have stronger performance.

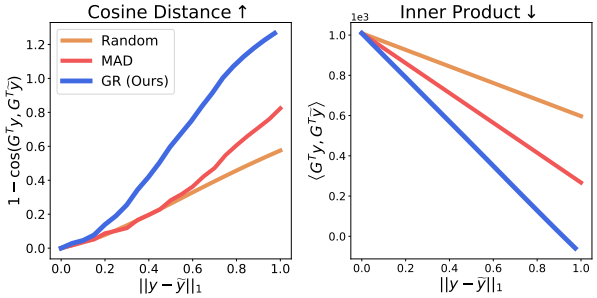

Figure 2. Compared to the previous state-of-the-art (MAD), gradient redirection (GR) results in larger perturbations to update gradients for a model stealing attack. MAD maximizes angular deviations with a heuristic algorithm (left), and GR minimizes inner product with a provably optimal algorithm (right). For both metrics, gradient redirection gives substantially more leverage.

Model Stealing Defenses. Defenses against model stealing fall into two complimentary approaches, which can be combined in a swiss cheese model to obtain robust protection (Hendrycks et al., 2021b). The first line of defense attempts to directly thwart model extraction attempts by returning modified or censored predictions from the API without significantly reducing performance for benign users. The simplest such defense is truncating posteriors or only outputting the predicted class (Tramèr et al., 2016). This defense is used in real prediction APIs, such as the OpenAI API and AI21 Studio where posterior probabilities are truncated to the top 2% and 0.03% of values, respectively. Unfortunately, this reduces utility to benign users, precluding certain applications, and is harmful to external transparency.

Rather than truncate posteriors, several recent works have investigated slightly perturbing posteriors to derail extraction attempts. Lee et al. (2019) introduce ambiguity into clean posteriors, which lowers the accuracy of the stolen model while preserving the defender's accuracy. Building on this intuition, Kariyappa & Qureshi (2020) train a misinformation network to predict incorrect posteriors. Similarly, Krishna et al. (2019) replace clean posteriors with harmful posteriors based on how anomalous the query is (Hendrycks et al., 2018). Note that this defense strategy assumes a setting where attackers have difficulty obtaining queries close to the defender's training distribution, which may not be true in practice.

The prediction poisoning approach introduced by Orekondy et al. (2020) demonstrates that posteriors can be optimally perturbed to derail the backward pass of an extraction attack, similar to adversarial examples for the forward pass (Szegedy et al., 2013). They choose perturbations to poison the adversary's update gradient by maximizing angular deviation with the clean gradient. This results in a strong

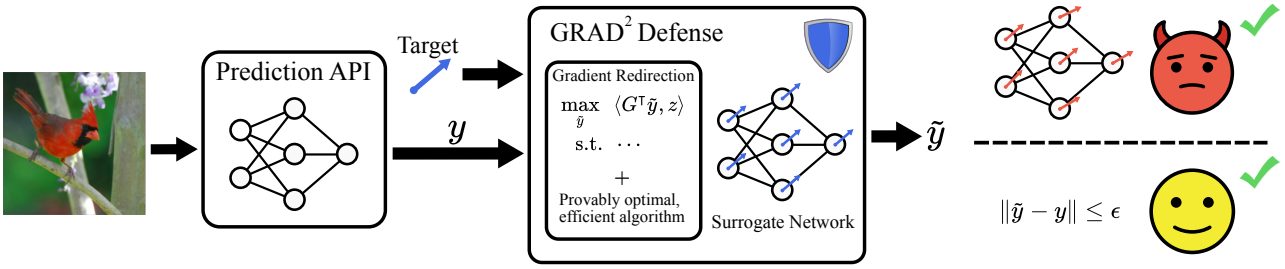

Figure 3. A user submits an image **left** to a prediction API, producing posteriors y. Our defense outputs an adjusted posterior $\tilde{y}$ to prevent a malicious user from stealing the model (**red, right**). The adjustment is chosen by an efficient, provably optimal, algorithm to produce the largest error in a surrogate network's gradient, which transfers to the unknown adversary network. Extensive experiments show that the stolen model has a significant loss in accuracy. Moreover, the benign user is happy, because the adjustments are guaranteed to be small.

defense, but it involves computing the full Jacobian of the network's posterior, an immense overhead cost that scales with the number of labels. For some applications, such as language modeling, the number of labels can be in the tens or hundreds of thousands. This is noted by Wallace et al. (2020), who focus on machine translation and adopt a best-of-k sampling procedure for selecting a perturbed prediction to derail the adversary's backward pass. However, this workaround still incurs an overhead of k samples and may lead to a suboptimal defense. Building on these works, we develop a new approach for optimally redirecting an adversary's gradient in a targeted and efficient manner.

Watermarking Defenses. A second overarching approach to model stealing defenses is digital watermarking. Many works purposefully insert backdoors into networks (Zhang et al., 2018; Adi et al., 2018), which enable proof of ownership. In the setting of model stealing, adversaries do not directly access model parameters and may be able to evade the watermark backdoor queries. Thus, different watermarking strategies are required to enable identifying models stolen through prediction APIs. Szyller et al. (2021) extend watermarks to this setting by outputting incorrect predictions for a small percentage of queried inputs. This enables the defender to prove ownership by querying the stolen model on these watermark examples. However, a downside of this defense is that the adversary knows its own training set, and thus has access to the watermarked inputs. In our investigation, we find that gradient redirection with strong surrogates enables black-box reprogramming of the adversary, which can insert watermarks into stolen models that are completely unknown to the adversary.

3. Threat Model

We consider interactions between a single attacker and defender. The defender allows users to access a deep neural network g through a prediction API, and the attacker attempts to steal the functionality of the defender's model through querying the API and training a clone model f on the resulting input-output pairs. The attacker's goal is to obtain high accuracy on the defender's test set.

3.1. Attacker's Strategy

The attacker chooses a query x and receives a prediction from the defender's network. To avoid detection defenses (Juuti et al., 2019; Chen et al., 2020), we assume the attacker sends queries that mimic benign queries. That is, we avoid adaptive querying strategies with unusual temporal structures (Papernot et al., 2017; Orekondy et al., 2019). In particular, we consider the Knockoff Nets attack without adaptive querying (Orekondy et al., 2019), which is similar to knowledge distillation (Hinton et al., 2015) and yields state-of-the-art attack performance (Orekondy et al., 2020; Kariyappa & Qureshi, 2020).

Knockoff Nets Attack. Let g be the defender's network, and let f be the attacker's clone model parametrized by $\theta \in \mathbb{R}^d$. The attacker chooses a dataset of queries $\mathcal{Q}$ beforehand and sends queries $x \in \mathcal{Q}$ in random order. Let $y = g(x)$ be the defender network's posterior on query x. The loss of the adversary's clone model f on the example (x, y) is $H(y, f(x)) = -\sum_i y_i \log f(x)_i$. The adversary's update gradient on this example is the negative gradient of the loss with respect to the parameters θ. Note that this update gradient can be written as

$$\begin{aligned} -\nabla_\theta H(y, f(x;\theta)) &= \sum_i y_i \nabla_\theta \log f(x;\theta)_i \\ &= y^\intercal G, \end{aligned}$$

where $G = \nabla_\theta \log f(x;\theta)$ is the $n \times d$ Jacobian matrix of the log-posteriors of f. Through collecting large numbers of queries and using standard techniques for training deep neural networks, the adversary can mount a successful extraction attack.

Query Distribution. An important variable in the adversary's attack is the choice of query dataset $\mathcal{Q}$. Prior work

uses query datasets with varying levels of similarity to the defender's training distribution, ranging from highly similar (Lee et al., 2019; Tramèr et al., 2016) to unrelated (Orekondy et al., 2020; Kariyappa & Qureshi, 2020). In some cases, collecting many queries similar to the defender's training distribution may be easy, and in other cases it may be challenging. Hence, we consider both possibilities in our threat model. We refer to adversaries as *distribution-aware* if they use queries sharing semantic content with the defender's training distribution. We refer to adversaries as *knowledge-limited* if they use queries with disjoint semantic content from the defender's training distribution.

3.2. Defender's Objective

The defender has no access to the adversary's parameters. Indeed, if the adversary mimics benign queries then the defender may not even know they are under attack. Thus, the challenge for the defender is to mitigate model stealing attempts at all times while preserving the utility of their API for benign users. This is typically accomplished by returning modified posteriors that make model stealing hard are representative of the clean posteriors.

A primary measure of utility to benign users is the defender's classification error on the test set. However, it is also important to consider the overall modification to returned posteriors, as benign users can derive substantial value from guarantees that the perturbed posteriors are close to the clean posterior. That is, a perturbed posterior $\widetilde{y}$ should satisfy $\|\widetilde{y} - y\|_1 \leq \epsilon$, where y is the clean posterior and ϵ is small. A good defense will reduce the accuracy of clone models while minimally increasing classification error on the test set and ℓ_1 distance to the clean posteriors.

4. Approach

A recent insight in the development of model stealing defenses is the notion of adversarial perturbations to the backward pass (Orekondy et al., 2020; Wallace et al., 2020). That is, perturbations to the defender's posterior can be designed to maximally change the attacker's update gradient. Unfortunately, existing methods for computing these perturbations are inefficient, requiring sample-based approaches or hundreds of backward passes for a single API query. Additionally, they lack flexibility as model stealing defenses, since they only seek to maximize angular deviation with the original update gradient. Here, we describe a new approach that we call gradient redirection, which pushes the attacker's update gradient in a target direction and is efficient to compute.

4.1. Gradient Redirection Problem

For simplicity, we start by assuming white-box access to the adversary's network f. We operationalize the defender's goal as minimally perturbing the defender's posterior y in order to maximally push the adversary's update gradient in a target direction $z \in \mathbb{R}^d$. For a given distillation example (x, y), we want to solve the optimization problem

$$\begin{aligned} \max_{\widetilde{y}} \quad & \langle G^\intercal \widetilde{y}, z \rangle \\ \text{s.t.} \quad & \mathbf{1}^\intercal \widetilde{y} = 1 \\ & \widetilde{y} \succeq 0 \\ & \|\widetilde{y} - y\|_1 \leq \epsilon, \end{aligned} \tag{1}$$

where y, $G = \nabla_\theta \log f(x; \theta)$, $z \in \mathbb{R}^d$, and $0 \leq \epsilon < 2$ are fixed. Typically we also have $y \in \Delta^{n-1}$, although for proofs we may have $\sum_i y_i < 1$. Note that this is a linear program, so in theory we could find the optimal $\widetilde{y}$ with performant LP solvers such as affine scaling variants of Karmarkar's algorithm (Adler et al., 1989). However, in real world cases, n may be in the tens of thousands, requiring upwards of 10GB just to store the constraint matrix for a single distillation example. As prediction APIs often handle high volumes of queries, solving this optimization problem with existing methods would be far too costly in practice.

Algorithm 1 Gradient Redirection

Input: G, z, y, ϵ
Output: $\widetilde{y}$
$\widetilde{y} \leftarrow y$
$s \leftarrow \text{argsort}(Gz)$
$\widetilde{y}_{s_n} \leftarrow \min(y_{s_n} + \epsilon/2,\ 1)$
$\lambda \leftarrow 0$
$t \leftarrow 1$
while $t < n$ **do**
  $\widetilde{y}_{s_t} \leftarrow \max(y_{s_t} - (\epsilon/2 - \lambda),\ 0)$
  if $y_{s_t} - (\epsilon/2 - \lambda) > 0$ **then**
    Return $\widetilde{y}$
  end if
  $\lambda \leftarrow \lambda + y_{s_t}$
  $t \leftarrow t + 1$
end while

4.2. Gradient Redirection Algorithm

We propose an efficient and provably optimal algorithm to solve the gradient redirection problem. First, we note that our problem is structurally similar to the fractional knapsack problem, which is solved in linearithmic time by a greedy algorithm (Dantzig, 2016). Drawing from this connection, we propose a knapsack-like greedy algorithm for solving gradient redirection in Algorithm 1. The inner product objective (2) can be rewritten as $\widetilde{y}^\intercal G z$, where $Gz \in \mathbb{R}^n$ can be interpreted as a value vector.

Our algorithm initializes $\widetilde{y}$ as y. It then proceeds by taking probability mass from indices of Gz with low values and putting as much mass as possible in the index of Gz with

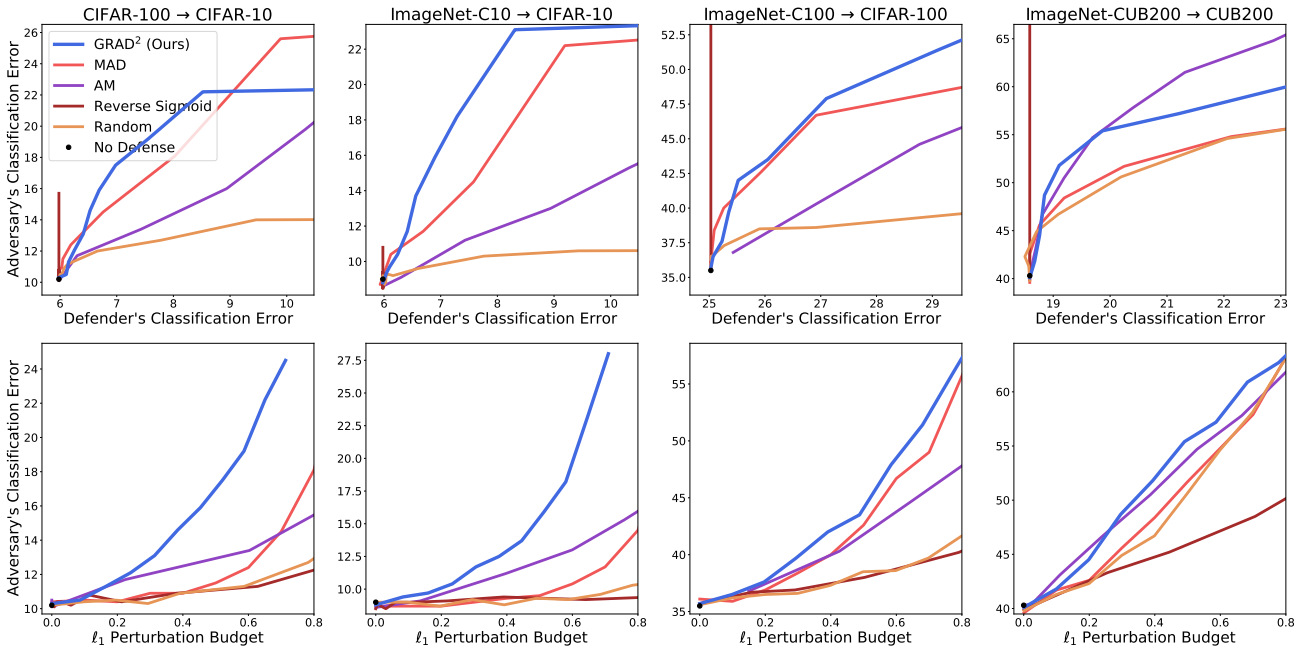

Figure 4. For most defense budgets, GRAD^2 induces higher classification error in the adversary than the best prior methods. While Reverse Sigmoid is efficient in terms of classification error, this performance requires an unreasonably large ℓ_1 perturbation budget, which renders the defense unusable in practice. By contrast, GRAD^2 has strong performance on both budget metrics.

the highest value. Intuitively, we are trading off less valuable indices for more valuable indices while taking care to respect the simplex constraint at each step. Our algorithm has two stopping conditions. In the first case, $\widetilde{y}_{s_n}$ attains $y_{s_n} + \epsilon/2$. This means that we added $\epsilon/2$ probability mass to $\widetilde{y}_{s_n}$. Hence, we removed $\epsilon/2$ probability mass from other indices of $\widetilde{y}$, so $|\widetilde{y} - y|_1 = \epsilon$, i.e. we hit the budget constraint. In the second case, we have $\widetilde{y}_{s_n} = 1$, i.e. we hit the simplex constraint. In both cases, we have moved as much mass as possible from the least valuable indices into the most valuable index.

Theorem 4.1. *Given a gradient redirection problem (G, z, y, ϵ) as formulated in (2), Algorithm 1 outputs a globally optimal solution in $\mathcal{O}(n \log(n))$ time.*

The proof follows the common practice for greedy algorithms of establishing the greedy choice property and optimal substructure for a hierarchy of subproblems. The theorem then follows by induction. Please see the Supplementary Material for a full proof.

Computing Gz With Double Backprop. Algorithm 1 is fast for individual distillation examples (x, y). However, it assumes that we are given $G = \nabla_\theta \log f(x; \theta)$, which itself requires n backward passes through f to compute. In model stealing defenses, f represents a neural network, potentially with millions of parameters. Hence, directly computing G is impractical. To obtain an optimal $\widetilde{y}$ when starting from the raw (x, y) pair, we need a way to circumvent this computational bottleneck.

We solve this problem with double backpropagation. Note that G is only used in Algorithm 1 to compute the matrix-vector product Gz, which is the gradient with respect to $\widetilde{y}$ of $\langle G^\intercal \widetilde{y}, z \rangle$. We know $G^\intercal \widetilde{y}$ is the gradient of a cross-entropy loss with respect to θ, since we have $G^\intercal \widetilde{y} = \sum_i \widetilde{y}_i \nabla_\theta \log f(x; \theta)_i = -\nabla_\theta H(\widetilde{y}, f(x; \theta))$. Thus, $G^\intercal \widetilde{y}$ can be computed with a single backward pass, and Gz can be computed as $-\nabla_{\widetilde{y}} z^\intercal \nabla_\theta H(\widetilde{y}, f(x; \theta))$ by backpropagating through the computation graph representing the first backward pass, a procedure known as double backpropagation that is supported in many machine learning frameworks. We denote the composition of our gradient redirection algorithm with double backprop as $\mathrm{GR}(f, x, y, z, \epsilon)$ for network f, query x, clean posterior y, target direction z, and perturbation budget ϵ. The cost of computing Gz with double backpropagation is roughly comparable to three additional forward passes on top of computing the output $f(x; \theta)$ (one for the first backward pass + two for double backprop), rather than n additional forward passes, as would be necessary when directly computing G.

4.3. Model Stealing Defense

We present a new model stealing defense called the Gradient Redirection Adversarial Distillation Defense, abbreviated GRAD^2. Our defense is based on the gradient redirection algorithm. However, since the attacker's network is unknown to the defender, we cannot directly apply the algorithm. Thus, we represent the adversary with a surrogate network

	ImageNet-C10 → CIFAR-10						ImageNet-C100 → CIFAR-100						ImageNet-CUB200 → CUB200					
	Δ Clf. Err			ℓ_1 Distance			Δ Clf. Err			ℓ_1 Distance			Δ Clf. Err			ℓ_1 Distance		
Method	1	2	5	0.1	0.2	0.5	1	2	5	0.1	0.2	0.5	1	2	5	0.1	0.2	0.5
Random	9.8	10.3	10.6	9.0	8.7	9.3	38.5	38.6	39.8	36.2	36.5	38.5	48.5	51.4	56.0	41.3	42.3	50.7
Reverse Sigmoid	-	-	-	9.0	9.1	9.3	-	-	-	36.3	36.8	38.0	-	-	-	41.2	42.6	45.9
Adaptive Mis.	10.4	11.9	16.3	9.0	9.6	12.1	38.2	40.6	46.6	36.4	37.4	41.8	53.8	**58.6**	**66.8**	**42.8**	**45.6**	53.8
MAD	12.6	16.4	22.6	8.7	8.7	9.5	43.0	46.8	49.2	35.9	36.9	42.6	49.6	52.3	56.0	41.7	42.6	51.7
GRAD^2 (Ours)	**16.4**	**21.5**	**23.4**	**9.5**	**10.1**	**15.5**	**43.4**	**47.6**	**53.0**	**36.5**	**37.7**	**44.1**	**54.1**	56.4	60.7	41.8	44.6	**55.6**

Table 1. For distribution-aware adversaries, GRAD^2 increases the adversary's classification error more than state-of-the-art baselines for nearly all budgets and test conditions. All values aside from ℓ_1 budgets are percentages. For fair comparison, dashes indicate cases where the ℓ_1 Distance metric is untenable; see text. **Bold** is best and underline is second-best.

h and design optimal perturbations for h instead.

Improved Surrogate Networks. Although prior works have used surrogate networks for model stealing defenses (Orekondy et al., 2020; Wallace et al., 2020), very little is known about how to design good surrogates. In particular, an important unanswered question is whether perturbations designed for the surrogate actually transfer to the attacker's network in the first place. To fill this gap in understanding, we conduct a detailed analysis of various design choices for surrogate networks, reporting results in the Appendix. Our main findings are 1) Perturbations designed for surrogate networks do in fact transfer to the attacker's network, 2) We can train the surrogate on the attacker's queries to obtain better transfer and a stronger downstream defense, and 3) Early stopping of surrogate training leads to a stronger downstream defense. The second finding is a crucial difference between our surrogates and those in previous work. Namely, we notice that the only information we have about the adversary is the query set $\mathcal{Q}$. Thus, to make the surrogate more representative of the adversary, we can train the surrogate on this data using knowledge distillation from the defender's network. In online scenarios, this requires continual learning. For simplicity, we assume the adversary sends all queries in a batch before beginning training.

Coordinated Defense. A key advantage of gradient redirection is the freedom of choosing a target direction z. We investigate several choices of z and discuss their properties. An intuitive choice is $z = \nabla_\theta H(y, h(x; \theta_h))$, where the target points opposite to the clean gradient. This setting of z is similar to the MAD algorithm from (Orekondy et al., 2020), which finds $\widetilde{y}$ obtaining a high cosine distance with the clean gradient $1 - \cos(\widetilde{y}^\intercal G, y^\intercal G)$. In Figure 2, we compare this setting of z to the MAD algorithm, finding that even though gradient redirection optimizes the inner product as opposed to cosine distance, we outperform MAD on both objectives. When incorporated into a full defense, perturbations from these defenses on two different examples may point in opposite directions and cancel out. Hence, we consider the possibility of coordinating the defense so that perturbations combine constructively rather than destructively.

Eval Data	Defender Accuracy	Attacker Accuracy	
		Knowledge-Limited	Distribution-Aware
CIFAR-10	94.0	89.8	91.0
CIFAR-100	75.0	55.5	64.5
CUB200	81.4	58.7	59.7

Table 2. Accuracy of the defender's classifier and stolen classifiers with no defense applied. Distribution-aware attacks are far more effective than knowledge-limited attacks and hence are more important to defend against.

The simplest possible coordinated defense uses $z = \mathbf{1}$, which pushes the attacker's parameters in the unhelpful direction of the all-ones vector. Since z is constant and does not depend on x, it has the desirable property of invariance to batching. That is, computing $\text{GR}(h, x, y, \mathbf{1}, \epsilon)$ in parallel on a batch of inputs gives the same batch of output posteriors whether one obtains Gz via per-example gradients or the batch gradient. In Figure 9, we show that coordinated defenses outperform uncoordinated ones.

Full GRAD^2 Method. Our full gradient redirection defense combines Algorithm 1 with our improved surrogates and coordinated defense strategy. Namely, we use a surrogate h trained on the adversary's queries $\mathcal{Q}$ with early stopping after $E = 10$ epochs, and we set $z = \mathbf{1}$.

5. Experiments

Datasets. We use three evaluation datasets: CIFAR-10, CIFAR-100, and CUB200 (Krizhevsky et al., 2009; Welinder et al., 2010). For each evaluation dataset, we explore knowledge-limited and distribution-aware adversaries. The knowledge-limited query sets for the above evaluation datasets are CIFAR-100, CIFAR-10, and Caltech-256 respectively (Griffin et al., 2007). For distribution-aware

Method	CIFAR-100 → CIFAR-10						CIFAR-10 → CIFAR-100						Caltech256 → CUB200					
	Δ Clf. Err			ℓ_1 Distance			Δ Clf. Err			ℓ_1 Distance			Δ Clf. Err			ℓ_1 Distance		
	1	2	5	0.1	0.2	0.5	1	2	5	0.1	0.2	0.5	1	2	5	0.1	0.2	0.5
Random	12.2	12.8	14.0	10.4	10.5	11.1	50.9	52.1	54.5	46.5	47.8	50.6	53.8	58.1	65.1	43.1	45.2	57.1
Reverse Sigmoid	-	-	-	10.7	10.5	11.1	-	-	-	46.0	46.9	50.8	-	-	-	42.7	44.2	49.7
Adaptive Mis.	12.7	14.3	21.7	**10.8**	11.5	12.9	47.6	51.0	60.2	**47.5**	**50.6**	**61.2**	**64.7**	**70.6**	-	43.3	45.6	53.4
MAD	15.1	18.0	**25.9**	10.5	10.4	11.5	52.2	53.6	58.6	45.1	46.7	52.0	55.4	57.7	62.1	**43.4**	**47.6**	57.1
GRAD^2 (Ours)	**17.5**	**20.5**	22.4	10.6	**11.7**	**17.0**	**55.2**	**59.3**	**63.7**	46.3	48.0	56.8	57.9	60.7	**65.2**	42.5	46.1	**58.3**

Table 3. For knowledge-limited adversaries, GRAD^2 increases the adversary's classification error more than state-of-the-art baselines in most test conditions. All values aside from ℓ_1 budgets are percentages. For fair comparison, dashes indicate cases where the ℓ_1 Distance metric is untenable; see text. **Bold** is best and underline is second-best.

adversaries, we construct query sets from ImageNet-1K by manually selecting overlapping classes (Deng et al., 2009). This gives us ImageNet-C10, ImageNet-C100, and ImageNet-CUB200, which are paired with their matching evaluation set and contain $183,763$, $161,653$, and $30,000$ examples respectively.

Training. Our experiments have three stages. In the first stage, a defender network trains on each evaluation dataset. In the second stage, defense methods generate protected posteriors for each query set and evaluation dataset at various defense budgets. Finally, adversary networks train on the protected posteriors. We denote experiments with transfer data $\mathcal{Q}$ and evaluation data $\mathcal{D}$ as $\mathcal{Q} \rightarrow \mathcal{D}$. On CUB200, we fine-tune ResNet50 networks pre-trained on ImageNet. For other datasets, we train 40-2 Wide ResNets from scratch. This allows us to gauge whether pre-training can substantially alter results. All networks are trained for 50 epochs using SGD with Nesterov momentum of 0.9. We use an initial learning rate of 0.01 for CUB200 and 0.1 for other evaluation datasets. The learning is annealed with a cosine schedule, and we use weight decay of $5 \cdot 10^{-4}$.

Metrics. We evaluate defenses with the adversary's classification error on the defender's test set for a given budget. Following Orekondy et al. (2020), we use two budget metrics: defender classification error and ℓ_1 distance between the modified posteriors and the clean posteriors averaged across the query set. For metrics in the tables, we report the increase in classification error. We denote these by "Δ Clf. Err" and "ℓ_1 Distance" respectively. A high value for either budget metric renders the defender's network unusable, so strong defenses will induce large adversary errors for small values of both budget metrics. Thus, we focus our comparisons on values of Δ Clf. Err and ℓ_1 Distance in a realistically acceptable range of trade-offs.

Baselines. We compare GRAD^2 to several baseline defenses. *No Defense:* The adversary trains on clean posteriors returned from the API. No attempt is made at defense. *Random:* The defender's posterior is interpolated with a 1-hot posterior, where the index of the nonzero entry is selected at random from all the labels that are different from the argmax prediction of the clean posterior. *Reverse Sigmoid* (Lee et al., 2019): The defender's posterior is modified with a noninjective analogue of the sigmoid function, making it challenging for the adversary to exactly recover the clean posterior. *Adaptive Misinformation (AM)* (Kariyappa & Qureshi, 2020): An out-of-distribution (OOD) detector flags anomalous queries as adversarial and interpolates with posteriors from a misinformation network that misclassifies the defender's training set. Note that this method assumes that attacker queries will be OOD, which is a fundamentally different approach from ours. However, we can still compare performance. *MAD* (Orekondy et al., 2020): A heuristic algorithm is used to find perturbed posteriors where the resulting perturbed gradient on a surrogate network has high angular deviation with the clean gradient. The surrogate network is randomly initialized.

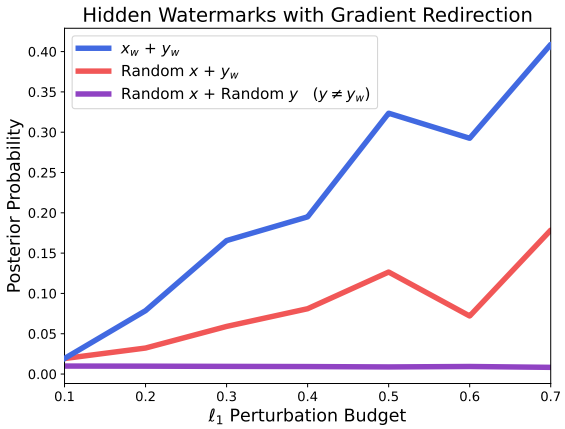

Figure 5. Gradient redirection enables reprogramming the adversary to predict a desired class on a hidden watermark image that only the defender knows. Posterior probabilities on non-target images and labels is lower than for the target input-output pair.

5.1. Comparing Defenses

We compare defenses at practical values of Δ Clf Err and ℓ_1 Distance and show the results in Tables 1 and 3. In Figure 4,

we visualize results. For a fairer comparison, tabular results are dashed out if ℓ_1 Distance is greater than 1.0, a conservatively large value. This is because the Δ Clf Err metric can be gamed by simply increasing the temperature of the posterior, which preserves classification error but destroys information. When considering both budget metrics, GRAD^2 matches or outperforms the best prior methods in most cases. Additionally, in terms of raw numbers GRAD^2 often outperforms other defenses by a significant margin. For example, at a classification error budget of 1%, our defense increases the error of an adversary seeking to steal a CIFAR-10 model with ImageNet-C10 queries from 9% to 16.4%, a 30% relative improvement over the next best method.

Balancing Both Budget Metrics. By examining both the classification error and ℓ_1 budgets of the defender, we see that methods which perform exceptionally well on one metric can perform very poorly on the other. Namely, Reverse Sigmoid is accuracy-preserving for a large range of hyperparameters, resulting in very high adversary error for minuscule increases in classification error budget. However, its ℓ_1 Distance budget scales more quickly than all other baselines, indicating that it would not be very useful in practice. By contrast, GRAD^2 has balanced performance on both budget metrics.

Robustness to Threat Model. We find that varying the threat model assumptions can significantly affect some defenses. In particular, Adaptive Misinformation relies on being able to identify queries from adversaries with out-of-distribution detectors, so it may perform less well if queries are closer to the defender's training distribution. Accordingly, the gap between GRAD^2 and AM is larger for the distribution-aware adversaries than for knowledge-limited adversaries. With a distribution-aware adversary, GRAD^2 is still successful with a small budget, demonstrating robustness to variations in the threat model.

5.2. Reprogramming The Adversary

As we have full control over the target direction z in Algorithm 1, a natural question is whether gradient redirection can be used for more than just increasing the adversary's error. We answer this in the affirmative by showing that in ideal conditions gradient redirection can reprogram adversaries to behave in a desired way on watermark images. Importantly, unlike prior work on watermarking defenses for prediction APIs (Szyller et al., 2021), we can select our watermark images at will. We are not restricted to using images that the adversary has sent as queries.

Let (x_w, y_w) be an input-output pair that we want the adversary f to predict. To demonstrate watermarking, we assume ideal conditions with white-box access to f. Let the gradient redirection target be $z = -\nabla_\theta H(y_w, f(x_w; \theta))$. This target is updated after every training step. We experiment

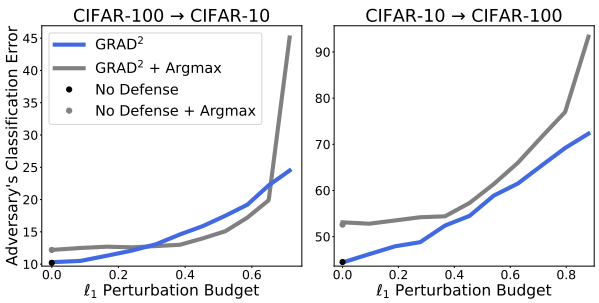

Figure 6. GRAD^2 remains a strong defense even when the adversary employs countermeasures. Training on the argmax label alone results in a higher error for the adversary in most settings, demonstrating the robustness of GRAD^2.

with knowledge-limited adversaries on the CIFAR datasets. For each experiment, we insert a single (x_w, y_w) watermark into f. We perform each experiment three times with different randomly-selected (x_w, y_w) pairs, where x_w is selected from the defender's test set, and we average results. In Figure 5, we plot the posterior probability $f(x_w)_{y_w}$ at various perturbation budgets for the converged stolen model f. We also plot the average value of $f(x)_{y_w}$ when given random inputs x from the test set and $f(x)_y$ for random x and y. Gradient redirection enables reprogramming the adversary to have abnormally high posteriors on the watermark input-output pair.

5.3. Adversary Countermeasures

To evaluate how adversary countermeasures affect our GRAD^2 defense, we train knowledge-limited adversaries on CIFAR-10 and CIFAR-100 with the argmax label of the perturbed posteriors rather than the entire posterior. In Figure 6, we plot results. We find that adversaries trained with this countermeasure obtain a higher initial error, although they can preserve accuracy in the face of more aggressive defenses. GRAD^2 remains a strong defense in the face of this countermeasure, with a higher adversary error in most cases, demonstrating the robustness of our approach.

6. Conclusion

We introduced gradient redirection, a new approach to model stealing defenses that enables modifying the adversary's update gradient in a targeted manner. We presented a provably optimal algorithm to efficiently solve gradient redirection problems, which we use to construct the GRAD^2 model stealing defense. In experiments, our defense outperformed all prior defenses and was robust to adversary countermeasures. Moreover, we showed that gradient redirection can be used to reprogram the adversary in a desired manner, which we hope will help foster further work on model stealing defenses.

Acknowledgements

This work is partially supported by NSF grant No.1910100, NSF CNS 20-46726 CAR, C3 AI, and the Sloan Fellowship.

References

Adi, Y., Baum, C., Cisse, M., Pinkas, B., and Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 1615–1631, 2018.

Adler, I., Resende, M. G., Veiga, G., and Karmarkar, N. An implementation of karmarkar's algorithm for linear programming. *Mathematical programming*, 44(1):297–335, 1989.

Buchanan, B., Lohn, A., Musser, M., and Sedova, K. Truth, lies, and automation. 2021.

Chen, S., Carlini, N., and Wagner, D. Stateful detection of black-box adversarial attacks. In *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*, pp. 30–39, 2020.

Dantzig, G. *26. Discrete-Variable Extremum Problems*. Princeton University Press, 2016.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Griffin, G., Holub, A., and Perona, P. Caltech-256 object category dataset. 2007.

Hendrycks, D., Mazeika, M., and Dietterich, T. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

Hendrycks, D., Burns, C., Chen, A., and Ball, S. Cuad: An expert-annotated nlp dataset for legal contract review. *NeurIPS*, 2021a.

Hendrycks, D., Carlini, N., Schulman, J., and Steinhardt, J. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*, 2021b.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Juuti, M., Szyller, S., Marchal, S., and Asokan, N. Prada: Protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS P)*, pp. 512–527, 2019. doi: 10.1109/EuroSP.2019.00044.

Kariyappa, S. and Qureshi, M. K. Defending against model stealing attacks with adaptive misinformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2020.

Krishna, K., Tomar, G. S., Parikh, A. P., Papernot, N., and Iyyer, M. Thieves on sesame street! model extraction of bert-based apis. *arXiv preprint arXiv:1910.12366*, 2019.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lee, T., Edwards, B., Molloy, I., and Su, D. Defending against neural network model stealing attacks using deceptive perturbations. In *2019 IEEE Security and Privacy Workshops (SPW)*, pp. 43–49, 2019. doi: 10.1109/SPW.2019.00020.

Orekondy, T., Schiele, B., and Fritz, M. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4954–4963, 2019.

Orekondy, T., Schiele, B., and Fritz, M. Prediction poisoning: Towards defenses against dnn model stealing attacks. In *ICLR*, 2020.

Pal, S., Gupta, Y., Shukla, A., Kanade, A., Shevade, S., and Ganapathy, V. Activethief: Model extraction using active learning and unannotated public data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 865–872, 2020.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

Radford, A., Wu, J., Amodei, D., Amodei, D., Clark, J., Brundage, M., and Sutskever, I. Better language models and their implications. *OpenAI Blog*, 1:2, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Szyller, S., Atli, B. G., Marchal, S., and Asokan, N. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 4417–4425, 2021.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 601–618, 2016.

Wallace, E., Stern, M., and Song, D. Imitation attacks and defenses for black-box machine translation systems. *arXiv preprint arXiv:2004.15015*, 2020.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Zhang, J., Gu, Z., Jang, J., Wu, H., Stoecklin, M. P., Huang, H., and Molloy, I. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18, pp. 159–172, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355766. doi: 10.1145/3196494.3196550.

A. Proving Optimality

A.1. Gradient Redirection Problem

For simplicity, we start by assuming white-box access to the adversary's network f. We operationalize the defender's goal as minimally perturbing the defender's posterior y in order to maximally push the adversary's update gradient in a target direction $z \in \mathbb{R}^d$. For a given distillation example (x, y), we want to solve the optimization problem

$$\begin{aligned} \max_{\widetilde{y}} \quad & \langle G^\intercal \widetilde{y}, z \rangle \\ \text{s.t.} \quad & \mathbf{1}^\intercal \widetilde{y} = 1 \\ & \widetilde{y} \succeq 0 \\ & \|\widetilde{y} - y\|_1 \leq \epsilon, \end{aligned} \tag{2}$$

where y, $G = \nabla_\theta \log f(x; \theta)$, $z \in \mathbb{R}^d$, and $0 \leq \epsilon < 2$ are fixed. Typically we also have $y \in \Delta^{n-1}$, although for establishing optimal substructure we may have that $\sum_i y_i < 1$. Note that this is a linear program reminiscent of knapsack problems.

A.2. Gradient Redirection Algorithm

We propose an efficient and provably optimal algorithm to solve the gradient redirection problem, which we describe in the main paper. First, note that the inner product objective (2) can be rewritten as $\widetilde{y}^\intercal G z$, where $Gz \in \mathbb{R}^n$ can be interpreted as a value vector. For brevity, let $c = Gz$, and let $s = \text{argsort}(c)$, i.e. $c_{s_1} \leq c_{s_2} \leq \cdots \leq c_{s_n}$.

Our algorithm initializes $\widetilde{y}$ as y. It then proceeds by taking probability mass from indices of c with low values and putting as much mass as possible in c_{s_n}, which has the highest value of all dimensions in c. Intuitively, we are trading off less valuable indices for more valuable indices while taking care to respect the simplex constraint at each step. Our algorithm has two stopping conditions. In the first case, $\widetilde{y}_{s_n}$ attains $y_{s_n} + \epsilon/2$. This means that we added $\epsilon/2$ probability mass to $\widetilde{y}_{s_n}$. Hence, we removed $\epsilon/2$ probability mass from other indices of $\widetilde{y}$, so $|\widetilde{y} - y|_1 = \epsilon$, i.e. we hit the budget constraint. In the second case, we have $\widetilde{y}_{s_n} = 1$, i.e. we hit the simplex constraint. In both cases, we have moved as much mass as possible from the least valuable indices into the most valuable index.

Lemma A.1 (Greedy Choice Property)**.** *Let (G, z, y, ϵ) be a gradient redirection problem as formulated in (2), but with the budget constraint changed to $|\widetilde{y} - y|_1 + \lambda \leq \epsilon$, where $\lambda = 1 - \sum_i y_i$ ($\lambda = 0$ in the original problem). Let y^* be an optimal solution. Then we must have (a): $y^*_{s_n} = \min(y_{s_n} + \epsilon/2,\, 1)$ and (b): $y^*_{s_1} = \max(y_{s_1} - (\epsilon/2 - \lambda),\, 0)$. Furthermore, if $y^*_{s_1} \neq 0$, then we have (c): $y^*_{s_t} = y_{s_t}$ for $1 < t < n$.*

Proof. First we will prove (a). Assume for contradiction that there is an optimal solution y° such that $y^\circ_{s_n} \neq \min(y_{s_n} + \epsilon/2,\, 1)$. Consider the first case, where $1 \leq y_{s_n} + \epsilon/2$. This implies $y^\circ_{s_n} < 1$. Intuitively, we can find a feasible solution v with $v_{s_n} = 1$ that obtains a higher objective value than y°. Let v be the posterior with $v_{s_n} = 1$ and $v_i = 0$ for $i \neq s_n$. In the present case, we have $1 - y_{s_n} \leq \epsilon/2$, which implies $|v - y|_1 + \lambda = (1 - y_{s_n}) + \sum_{1 \leq t < n} y_{s_t} + (1 - \sum_i y_i) = 2(1 - y_{s_n}) \leq \epsilon$. Thus, v is a feasible solution. Since y° and v both sum to 1, we know v has a higher objective value than y°, which is a contradiction.

Now consider the second case, where $1 > y_{s_n} + \epsilon/2$. This implies $y^\circ_{s_n} < y_{s_n} + \epsilon/2 < 1$. Intuitively, we can find a feasible solution v that obtains a higher objective value than y° by moving mass into index s_n. Let I be the indices where $y^\circ_i \geq y_i$, and let J be the indices where $y^\circ_j < y_j$. We have $|y^\circ - y|_1 + \lambda = (\sum_{i \in I} y^\circ_i - y_i) + (\sum_{j \in J} y_j - y^\circ_j) + 1 - \sum_i y_i = 1 + (\sum_{i \in I} y^\circ_i - 2y_i) - \sum_{j \in J} y^\circ_j = 2\sum_{i \in I} y^\circ_i - y_i$, so we have $\sum_{i \in I} y^\circ_i - y_i \leq \epsilon/2$. Suppose we have $|y^\circ - y|_1 + \lambda < \epsilon$. Then we can just increase $y^\circ_{s_n}$ and decrease other entries of y° to maintain the simplex constraint until we hit the budget constraint, which would yield a solution with a higher objective value. Now suppose we have $|y^\circ - y|_1 + \lambda = \epsilon$. This implies $\sum_{i \in I} y^\circ_i - y_i = \epsilon/2$. In other words, the indices where $y^\circ_i > y_i$ account for a difference of $\epsilon/2$, but they are not concentrated in $y^\circ_{s_n}$ by our assumption. Simply move them to $y^\circ_{s_n}$ to obtain another feasible solution with greater objective value than that of y°. This is a contradiction for the second case, so we have proven (a).

Now we will prove (b). In the case where $1 \leq y_{s_n} + \epsilon/2$, we know $y^*_{s_n} = 1$ and $y^*_i = 0$ for $i \neq s_n$ from part (a). We also have $y_{s_1} - (\epsilon/2 - \lambda) < (\sum_{i \neq s_n} y_i) - \epsilon/2 + 1 - \sum_i y_i = 1 - \epsilon/2 - y_{s_n} < 0$, so $\max(y_{s_1} - (\epsilon/2 - \lambda),\, 0) = 0$, and we know $y^*_{s_1} = 0$. Hence, (b) is true in the case where $1 \leq y_{s_n} + \epsilon/2$.

Now consider the case where $1 > y_{s_n} + \epsilon/2$. Assume for contradiction that there is an optimal solution y° such that $y^\circ_{s_1} \neq \max(y_{s_1} - (\epsilon/2 - \lambda), 0)$. By (a), we know $y^\circ_{s_n} = y_{s_n} + \epsilon/2$. Let I be the indices where $y^\circ_i > y_i$, and let J be the indices where $y^\circ_j \leq y_j$. By analogous argument to that in part (a), we have $\sum_{i \in I} y^\circ_i - y_i = \epsilon/2$. That is, indices where $y^\circ_i > y_i$ account for a difference of exactly $\epsilon/2$. But this means that s_n is the only such index, so we have $y^\circ_j \leq y_j$ for $j \neq s_n$. Furthermore, we have $|y^\circ - y|_1 + \lambda = (\sum_{i \in I} y^\circ_i - y_i) + (\sum_{j \in J} y_j - y^\circ_j) + \lambda = \epsilon/2 + (\sum_{j \in J} y_j - y^\circ_j) + \lambda = \epsilon$, so we have $(\sum_{j \in J} y_j - y^\circ_j) = \epsilon/2 - \lambda$. This means that the indices $j \neq s_n$ account for a difference of exactly $\epsilon/2 - \lambda$. By concentrating this difference in $y^\circ_{s_1}$, we can improve the objective value while maintaining a feasible solution. If $y_{s_1} - (\epsilon/2 - \lambda) \geq 0$, we will be able to concentrate all this difference into $y^\circ_{s_1}$, in which case $y^\circ_{s_t} = y_{s_t}$ for $1 < t < n$. If $y_{s_1} - (\epsilon/2 - \lambda) < 0$, we will not be able to concentrate all of the difference into $y^\circ_{s_1}$ due to the nonnegativity constraint. However, concentrating as much of the difference as possible by setting $y^\circ_{s_1} = 1$ will still give a feasible solution with an improved objective value. This is a contradiction, so we have proven (b).

Now we will prove (c). Suppose y^* is an optimal solution with $y^*_{s_1} \neq 0$. In the proof of part (b), we saw that this only happens when we have $y_{s_1} - (\epsilon/2 - \lambda) \geq 0$. Furthermore, in this case we also have $y^*_{s_t} = y_{s_t}$ for $1 < t < n$, because we had to move all the remaining difference between y^* and y (outside of indices s_n and s_1) into decreasing $y^*_{s_1}$ as much as possible. This proves (c).

□

Lemma A.2 (Optimal Substructure)**.** *Let (G, z, y, ϵ) be a gradient redirection problem with the modified budget constraint from Lemma A.1. Namely, $|\widetilde{y} - y|_1 + \lambda \leq \epsilon$, where $\lambda = 1 - \sum_i y_i$. Let y^* be an optimal solution with $y^*_{s_1} = 0$, and let $s = \text{argsort}(Gz)$. Consider the subproblem (G', z', y', ϵ), where G' has row s_1 removed, z' and y' have index s_1 removed. Then y^* with index s_1 removed is an optimal solution to the subproblem. (Note that in the subproblem, y may lie outside the simplex, but the simplex constraint for $\widetilde{y}$ is unchanged.)*

Proof. Let $y^{*\prime}$ denote y^* with index s_1 removed. We know $\lambda' = 1 - \sum_i y'_i = \lambda + y_{s_1}$, and we know $\sum_i y^{*\prime}_i = \sum_i y^*_i = 1$. Thus, we have $|y^{*\prime} - y'|_1 + \lambda' = |y^* - y|_1 - \left|y^*_{s_1} - y_{s_1}\right| + \lambda + y_{s_1} = |y^* - y|_1 + \lambda \leq \epsilon$. Therefore, $y^{*\prime}$ is a feasible solution to the subproblem (G', z', y', ϵ). Assume for contradiction that there is a solution y° to the subproblem with a higher objective value than $y^{*\prime}$. Insert 0 into position s_1 of y° to create a solution to the original problem with the same objective value. Inserting 0 into position s_1 of $y^{*\prime}$ also creates a solution to the original problem with the same objective value. In fact, it recreates y^*. But y^* is an optimal solution to the original problem, so its objective value cannot be lower than that of the expanded y°. This is a contradiction, so our assumption must be false, which means that $y^{*\prime}$ is optimal for the subproblem. □

Theorem A.3. *Given a gradient redirection problem (G, z, y, ϵ) as formulated in (2), Algorithm 1 outputs a globally optimal solution in $\mathcal{O}(n \log(n))$ time.*

Proof. By part (a) of Lemma A.1, we know that the initialization steps of Algorithm 1 set $\widetilde{y}_{s_n}$ to the value of the optimal solution. If $\widetilde{y}_{s_n}$ is set to 1, then the while loop will proceed until its stopping condition without altering $\widetilde{y}$ at other indices, returning an optimal $\widetilde{y}$. Now consider the case where $\widetilde{y}_{s_n}$ is set to a value less than 1, then we know y has at least $\epsilon/2$ mass at indices other than s_n. This means $\lambda + y_{s_t}$ will eventually exceed $\epsilon/2$, at which point the if-then condition will be entered and the algorithm will return.

We proceed by induction, starting where the if-then condition is entered and continuing backwards through the while loop to the first step. Note that when the if-then condition is entered, we have just set $\widetilde{y}_{s_t}$ to $y_{s_t} - (\epsilon/2 - \lambda)$. Let (G', z', y', ϵ) be the subproblem with s_j indices removed for $j < t$. In this subproblem, $\widetilde{y}_{s_t}$ is replaced by $\widetilde{y}'_{s_1}$. From part (b) of Lemma A.1, we know $\widetilde{y}'_{s_1}$ is set to an optimal value for this subproblem. Moreover, from part (c) and the early exit step of the algorithm, we know that the entirety of $\widetilde{y}'$ returned at this point is an optimal solution to the subproblem, not just the values at indices s_1 and s_{n-t}. This gives us a base case. For the induction step, assume that $\widetilde{y}'$ is optimal at step $t > 1$ of the while loop. We want to show that $\widetilde{y}$ is optimal for the subproblem (G, z, y, ϵ) at step $t - 1$ of the while loop. By part (b) of Lemma A.1, we know that $\widetilde{y}'_{s_1}$ is set to an optimal value in the previous step of the while loop. By Lemma A.2, we know that simply appending this value to the optimal solution of the subproblem at step t of the while loop gives us an optimal solution to the subproblem at step $t - 1$. By induction, the complete $\widetilde{y}$ returned by Algorithm 1 is an optimal solution to the original gradient redirection problem.

Asymptotically, the most expensive step is the sorting operation. All other steps are local comparisons and operations on pairs of array elements or scalars, and hence have $\mathcal{O}(n)$ time complexity. Thus, the overall time complexity is $\mathcal{O}(n \log(n))$. □

B. Additional Experiments

In Figures 8 and 9, we analyze the effect of our design choices for the GRAD^2 defense. Namely, we show how the surrogate ability to transfer to the adversary makes a large difference in downstream performance. In Figure 9, we plot the performance of a coordinated defense and un-

coordinated defense. These analyses identify ways to build stronger defenses which future work could build on. Additionally, they highlight the general importance of surrogate networks and destructive interference, which prior work did not investigate in detail.

B.1. Improved Surrogate Networks

We analyze the effect of design choices for the surrogate network. Namely, we show how the surrogate network's ability to transfer to the adversary makes a large difference in the downstream performance of gradient redirection defenses. Let f and h be an adversary and surrogate network, respectively. Let θ be the parameters of f that the adversary trains during a model stealing attack. We compute Transfer Performance for the surrogate network h at perturbation budget ϵ as

$$\frac{1}{|\mathcal{Q}|}\sum_{x\in\mathcal{Q}}\cos\left(\widetilde{y}^T\nabla_\theta \log f(x;\theta), z\right) - \cos\left(y^T\nabla_\theta \log f(x;\theta), z\right)$$

where y is the defender model's output for query x, z is the gradient redirection target, and $\widetilde{y} = \text{GR}(h, x, y, \mathbf{1}, \epsilon)$ is the output of gradient redirection on the surrogate network. Transfer Performance measures the increase in cosine similarity between the perturbed gradient and the target direction z compared to the negative control of the cosine similarity between the clean gradient and z. Thus, Transfer Performance greater than zero indicates successful transfer, and values less than or equal to zero indicate failed transfer. While any gradient redirection target could be used, we focus on the all-ones target $z = \mathbf{1}$ in this analysis.

Setup. The surrogate network analysis is in Figure 8. We train surrogates using the Knockoff Nets objective with the query distribution $\mathcal{Q}$ (sold lines) and the defender's training distribution $\mathcal{D}$ (dashed lines). In both cases, we train surrogates for 50 epochs and perform early stopping after E epochs, where $E \in \{0, 10, 20, 30, 40\}$. We then train three independent adversaries for 50 epochs each, saving snapshots at each epoch. We compute Transfer Performance for a surrogate at each epoch of the adversary's training, averaging over the three independent adversary training runs. We also compute the converged adversary's classification error when using the surrogates in a GRAD^2 defense.

Training Surrogates on the Query Distribution. The only concrete information the defender has about an adversary is the query distribution. A natural question is whether we can leverage this information to build a stronger defense. In the top row of Figure 8, we see that Transfer Performance is higher for surrogates trained on $\mathcal{Q}$. In the bottom row, we see that this translates to a stronger downstream defense for trained surrogates. This suggests that better transfer from the surrogate to the adversary is in fact valuable for the downstream defense. Note that prior works found that trained surrogates yield weaker defenses than random surrogates (Orekondy et al., 2020), which may be an artifact of training the surrogates on $\mathcal{D}$ instead of $\mathcal{Q}$. We show that training on $\mathcal{Q}$ causes trained surrogates to outperform random surrogates by a significant margin.

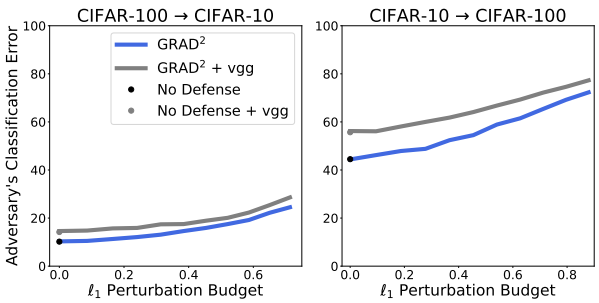

Figure 7. Posterior perturbations designed for a particular surrogate remain effective when the adversary uses a different network architecture.

Early Stopping of Surrogates. In the top row of Figure 8, we can see that surrogates that train for longer transfer better to adversaries at later stages of training, and early stopping of surrogates yields better transfer to adversaries at the initial stages of training. This provides further evidence that similarity to the adversary affects transfer and raises the question of whether early transfer or late transfer is more advantageous for a downstream defense. For both CIFAR-10 and CIFAR-100, we find that early stopping at $E = 10$ epochs yields the strongest defense in most cases, while values of $E \in \{20, 30, 40\}$ perform similarly. This suggests that early transfer is more important than late transfer.

B.2. Runtime Comparison

In Table 4, we compare the runtime of MAD and GRAD^2 on queries from CIFAR-10, CIFAR-100, and CUB200. We report the number of seconds to generate a perturbed posterior averaged across the test set. In all cases, GRAD^2 is substantially faster, and relative performance scales with the number of labels; GRAD^2 is $3.75\times$ faster on CIFAR-10, $4.32\times$ faster on CIFAR-100, and $6.33\times$ faster on CUB200.

B.3. Robustness Across Architectures

An important practical consideration for using gradient redirection defenses is whether surrogates remain effective if the adversary uses a different neural network architecture than the surrogate. In Figure 7, we evaluate GRAD^2 against adversaries using a different architecture than the standard surrogate used throughout the paper. In particular, we keep the same WRN-40-2 surrogate and posterior perturbations from the main experiments, but we change the adversary's architecture to VGG-16. We find that GRAD^2 remains a

Method	CIFAR-10	CIFAR-100	CUB200
MAD	0.15	1.21	2.66
GRAD^2	0.04	0.28	0.42

Table 4. Average time in seconds to generate a perturbed posterior for a single query on an NVIDIA A40 GPU. GRAD^2 is significantly faster than MAD.

strong defense in this setting, with almost no decrease in slope to the performance profile. This suggests that surrogates do transfer to different architectures.

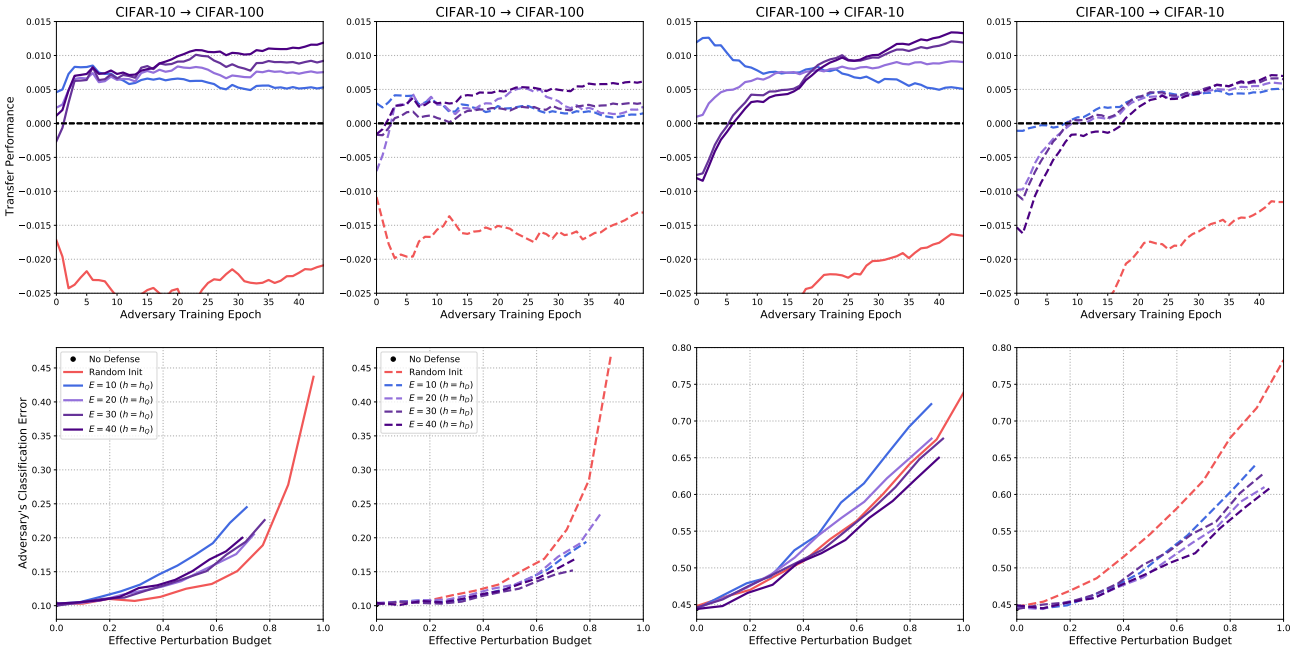

Figure 8. Surrogate analysis experiments. Top: Surrogates do transfer to adversaries with unknown parameters; Surrogates with early stopping after a small number of epochs transfer better to adversaries early in their training, whereas surrogates with early stopping after a larger number of epochs transfer better to adversaries later in their training; Surrogates trained with the query distribution $\mathcal{Q}$ (solid lines) transfer more effectively than surrogates trained with the defender's training distribution $\mathcal{D}$ (dashed lines); Random surrogates transfer very poorly (red lines). Bottom: Surrogate transfer matters, as stronger transfer results in a stronger downstream defense (higher adversary error for a given ℓ_1 budget); Surrogates with early stopping at low epochs ($E = 10$) yield a stronger downstream defense across datasets (blue lines); Randomly initialized surrogates (red lines) yield in a relatively weaker downstream defense, which surrogates trained on $\mathcal{Q}$ outperform. Notably, the MAD method uses randomly initialized surrogates, which transfer very poorly and underperform our surrogates trained on $\mathcal{Q}$ with early stopping at $E = 10$, which are used in the GRAD^2 defense.

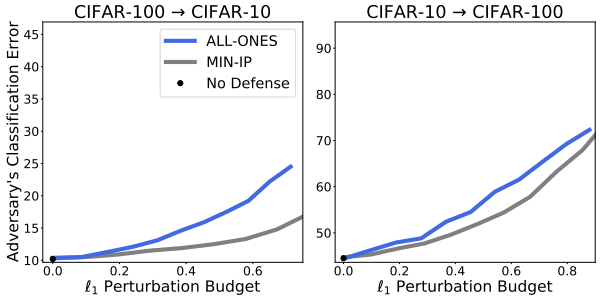

Figure 9. When we set the gradient redirection target z to the all-ones vector, this results in a defense that perturbs the adversary in a coordinated manner, providing that the perturbations transfer from the surrogate to the adversary. Perturbing away from the clean gradient, i.e. minimizing the inner product, may result in destructive interference of perturbations across the training set. Here, we measure the performance of these two choices of z, labeled ALL-ONES and MIN-IP respectively, where MIN-IP corresponds to $z = \nabla_\theta H(y, h(x; \theta_h))$ pointing opposite the clean. ALL-ONES corresponds to $z = \not\Vdash$, which does not depend on x and thus has a coordinated effect across training batches. We find that our coordinated defense outperforms the uncoordinated defense by a substantial margin.